

# Lecture 6: If-Else-If and Loops

Last time:

1. Finish Scanner
2. if statements

Today:

1. More on if
2. Project assigned
3. Named constants in Java
4. Loops





# Dangle Else: IfElseExample1

```
public class IfElseExample1 {  
  
    public static void main(String[] args) {  
        int x = 4;  
  
        if (x < 3)  
            if (x == 2)  
                System.out.println("X");  
        else  
            System.out.println("Y");  
    }  
}
```

# The “Dangling Else” Problem

- Which “if” an “else” is associated with can be ambiguous!
- Java rule: else is associated with “innermost” possible if
- Good programming practice: when in doubt, use { ... }
- **WE WILL USE { ... } FOR ALL IF, IF-ELSE, IF-ELSE-IF, STATEMENTS**

# Dangle Else: IfElseExample1 - Fixed



```
public class IfElseExample1 {  
  
    public static void main(String[] args) {  
        int x = 4;  
  
        if (x < 3) {  
            if (x == 2) {  
                System.out.println("X");  
            }  
        } else {  
            System.out.println("Y");  
        }  
    }  
}
```

# Cascading Elses

- A common programming paradigm:
  - “if something is true, do one thing”
  - “otherwise, if something else is true, do another thing”
  - “otherwise, if something else entirely is true, do yet another thing”
  - “otherwise, take a default action”
- How might we program this?



# IfElseExample2

```
public class IfElseExample2 {  
  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        System.out.print("Choose a number from 1 to 4: ");  
        int choice = scanner.nextInt();  
        String nameOfValue;  
        if (choice == 1) {  
            nameOfValue = "ONE";  
        }  
        else {  
            if (choice == 2) {  
                nameOfValue = "TWO";  
            }  
            else {  
                if (choice == 3) {  
                    nameOfValue = "THREE";  
                }  
                else {  
                    nameOfValue = "FOUR";  
                }  
            }  
        }  
        System.out.println("You chose " + nameOfValue);  
    }  
}
```

# Ugly and Confusing!

- Too many lines with only one }
- Easy to get lost in indentation
- However, we can use Java's "innermost if" rule for elses to program more clearly!



# IfElseExample2BetterVersion

```
public class IfElseExample2BetterVersion {  
  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        System.out.print("Choose a number from 1 to 4: ");  
        int choice = scanner.nextInt();  
        String nameOfValue;  
        if (choice == 1)  
            nameOfValue = "ONE";  
        else if (choice == 2)  
            nameOfValue = "TWO";  
        else if (choice == 3)  
            nameOfValue = "THREE";  
        else  
            nameOfValue = "FOUR";  
        System.out.println("You chose " + nameOfValue);  
    }  
}
```

**VIOLATES THIS RULE:**  
**WE WILL USE { ... } FOR ALL IF,  
IF-ELSE, IF-ELSE-IF, STATEMENTS**



# IfElseExample2BestVersion

```
public class IfElseExample2BetterVersion {  
  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        System.out.print("Choose a number from 1 to 4: ");  
        int choice = scanner.nextInt();  
        String nameOfValue;  
        if (choice == 1) {  
            nameOfValue = "ONE";  
        } else if (choice == 2) {  
            nameOfValue = "TWO";  
        } else if (choice == 3) {  
            nameOfValue = "THREE";  
        } else {  
            nameOfValue = "FOUR";  
        }  
        System.out.println("You chose " + nameOfValue);  
    }  
}
```

# A Common Programming Idiom



- “Idiom” = “convention”

- **if** (C1) {

S1;

} **else if** (C2) {

S2;

...

} **else** {

S<sub>n</sub>;

}

- Note indentation and curly bracket conventions!



# Project #1 Is Assigned

- You may see the assignment on the CMSC 131 web-site (click “Projects” link).
- It is due Sunday, **2/11 at 11 pm**
- The project is **open**:
  - You may discuss solution ideas with other students as well as TAs / instructors / etc.
  - You should document outside help using Java comments at top of your solution file
  - No code-copying allowed
  - Full policy is available from web-site

# Hints for Success

- Start now!
- Read entire assignment from beginning to end before starting to code
- Check out assignment from CVS now rather than later
- Follow the instructions *exactly*, as much of grading is automated

# In Project #1 Description

- You must use **meaningful variable names** and good indentation.
- You must use "**named constants**" for any literal values that will not change during program execution.

???

# Variable Name Conventions

- What is legal for variable names?
  - Letters, digits, \$, \_
  - Can't start variable name with digit
  - Avoid reserved words
- Use camel Case:
  - Variables & Methods use lower-case for first letter
  - Classes/Interfaces use upper-case for first letter

# Variable Name Conventions: Examples



- **Naming Conventions:** Standards developed over time.  
**Variables and methods:** Start with lowercase, and use uppercase for each new word:

`dataList2 myFavoriteMartian showMeTheMoney`

**Class names:** Start with uppercase and uppercase for each new word:

`String JOptionPane MyFavoriteClass`

**Named constants** (variables whose value never changes): All uppercase with underscores between words:

`MAX_LENGTH DAYS_PER_WEEK BOILING_POINT`

- Make variable names not too long, not too short
  - **Bad:** `crtltm`
  - **Bad :** `theCurrentItemBeingProcessed`
  - **Good :** `currentItem`

# Meaningful Variable Names

- Choose names for your variables to reflect their purpose

- Bad

```
String string = "";  
System.out.println ("Enter name:  ");  
string = sc.next();  
if (string.equals ("John Doe")) ...
```

- Good

```
String name = "";  
System.out.println ("Enter name:  ");  
name = sc.next();  
if (name.equals ("John Doe")) ...
```

# Named Constants in Java

- Programs often contain **literals** (= values)

e.g.

```
if (temp >= 97 && temp <= 99) ...
```

e.g.

```
System.out.print ("Enter integer: ");
```

- If same value should be used in several places, how to ensure consistency?
  - Check on temperature may be performed more than once
  - Same prompt might be printed in several places
- Java answer: **named constants**

# Named Constants

- `final int MAX_OK_TEMP = 99;`
  - Just like a regular variable declaration, except...
  - Special term `final`
  - Necessity of initial value
  - Any variable name will work, but convention is to use all capitals
- Difference with regular values: assignment attempt leads to error!

# Examples

- ```
final int MIN_OK_TEMP = 97;
final int MAX_OK_TEMP = 99;
...
if (temp >= MIN_OK_TEMP && temp <=
MAX_OK_TEMP) ...
```
- ```
final String INT_PROMPT = "Enter integer: ";
...
System.out.print (INT_PROMPT);
```

# Loops in Java

- So far our programs execute every program statement at most once
- Often, we want to perform operations more than once:
  - “Sum all numbers from 1 to 10”
  - “Repeatedly prompt user for input”
- Loops allow statements to be executed multiple times. Loop types in Java:
  - while
  - do-while
  - for
- We will study while, do-while now, for-loop later

# while and do-while Loops

- **while** and **do-while** loops contain:
  - A statement, called the **body**
  - A boolean **condition**
  - Idea: the body is executed as long as the condition is true
- **while-loop**: The condition is tested before each body execution

```
while ( <condition> )  
    <body>
```
- **do-while-loop**: The condition is tested after each body execution

```
do  
    <body>  
while ( <condition> );
```
- **Main difference**: do-while loop bodies always executed at least once

# Example 11

```
public class Example11 {  
  
    public static void main(String[] args) {  
        int i = 1;  
        while (i <= 10) {  
            System.out.println(i);  
            i = i + 1;  
        }  
    }  
}
```

“Loop counter”

Increment of loop counter ensures progress toward **loop termination**

# Infinite Loops

- Loops can run forever if condition never becomes false
- Be careful when programming loops!
  - Add statements for termination into loop body first
  - Make sure these statements are at end of body
  - e.g.

```
while (i <= 10) {  
    System.out.println(i);  
    i = i + 1;  
}
```

# Example 11b: Lots of Looping

```
public class Example11b {  
  
    public static void main(String[] args) {  
        int i = 1;  
        long total = 0;  
  
        while (i <= 1000000) {  
            total = total + i;  
            i = i + 1;  
        }  
  
        System.out.println("Total is: " + total);  
    }  
}
```

# Example 12: do-while

```
public class Example12 {  
  
    public static void main(String[] args) {  
        int i = 1;  
        do {  
            System.out.println(i);  
            i = i + 1;  
        } while(i <= 10);  
    }  
}
```

# Variables, Blocks and Scoping

- Variables can be declared anywhere in a Java program
- When are the declarations active?
  - After they are executed
  - *Only inside the block in which they are declared*
- **Scope rules** formalize which variable declaration are active when
  - **Global variables**: scope is entire program
  - **Local variables**: scope is a block

# Example 13

```
import java.util.Scanner;

public class Example13 {

    public static void main(String[] args) {
        int i = 1;
        Scanner sc = new Scanner(System.in);

        do {
            System.out.print("Enter an integer from 1 to 10: ");
            int answer = sc.nextInt();
        } while (answer < 1 || answer > 10);
        System.out.println("Good job.");
    }
}
```

What is scope of this declaration?

# Nested Loops

- while, do-while are statement constructors (like if-else, blocks)
- Loops can thus be used inside other loops!

# Example 14

```
public class Example14 {  
  
    public static void main(String[] args) {  
  
        int rowNumber = 1;  
        while (rowNumber < 10) {  
            int colNumber = 1;  
            while (colNumber < 10) {  
                System.out.print((rowNumber + colNumber) % 2);  
                colNumber = colNumber + 1;  
            }  
            System.out.println();  
            rowNumber = rowNumber + 1;  
        }  
    }  
}
```

Inner loop

Outer loop